Beyond Surveys: Analyzing Software Development Artifacts to Assess Teaching Efforts

Christoph Matthies, Ralf Teusner, Guenter Hesse
Hasso Plattner Institute
University of Potsdam, Germany
{firstname.lastname}@hpi.de

***Abstract*—This Innovative Practice Full Paper presents an approach of using software development artifacts to gauge student behavior and the effectiveness of changes to curriculum design. There is an ongoing need to adapt university courses to changing requirements and shifts in industry. As an educator it is therefore vital to have access to methods, with which to ascertain the effects of curriculum design changes. In this paper, we present our approach of analyzing software repositories in order to gauge student behavior during project work. We evaluate this approach in a case study of a university undergraduate software development course teaching agile development methodologies. Surveys revealed positive attitudes towards the course and the change of employed development methodology from Scrum to Kanban. However, surveys were not usable to ascertain the degree to which students had adapted their workflows and whether they had done so in accordance with course goals. Therefore, we analyzed students' software repository data, which represents information that can be collected by educators to reveal insights into learning successes and detailed student behavior. We analyze the software repositories created during the last five courses, and evaluate differences in workflows between Kanban and Scrum usage.**

***Index Terms*—software engineering, capstone course, development artifacts, Kanban, Scrum, Educational Data Mining**

I. Introduction

One of the main goals of universities is to provide students with an education of the best quality possible. As such, there is the constant need to improve the learning experience in courses, update course contents to changing requirements and strive for more effective organization structures [1]. With the continuing rise of digitization in universities, an ever-expanding amount of data on learners is available [2], [3]. Access to new data sources has led to drastic changes both in science and business. Equally, learning scientists can greatly benefit from having large repositories of educational data available [3]. Analysis of these repositories to tackle educational research issues has given rise to the field of Educational Data Mining (EDM) [4]. In this paper, we show how techniques from this domain can be used to gain insights into students' workflows and ascertain whether changes in curriculum design had the desired effects, without relying on traditional surveys alone.

A. Educational Data Mining

The International Educational Data Mining Society defines EDM as a discipline concerned with "developing methods for exploring the unique and increasingly large-scale data that come from educational settings and using those methods to better understand students, and the settings which they learn in" [5]. Traditional student assessment and evaluation methods such as standardized exams can only provide information on specific student traits at certain points in time. To obtain information that can explain students' progress, continued recording of their activities using more sophisticated techniques is necessary. If designed well, such measurements can provide insights into how students behave, communicate, and participate in learning activities [6]. In an educational setup, data is generated using a variety of disparate systems [2]. Some settings in which EDM has been applied include:

- E-learning and learning management systems, e.g. Moodle [7], where web mining approaches have been applied to student data in log files and databases [4].
- Massive Open Online Courses (MOOCs), where student collaborations and interactions with the platforms are studied [8].
- Offline education, where students are lectured in a traditional face-to-face manner. Statistical analyses are applied to students' data, like test scores or peer assessments, which are gathered in classroom environments [4].

Analyzing this educational data can help evaluate, validate and eventually improve courses and educational systems, paving the way for a more effective learning process [9].

B. Educational Data Sources

Educational contexts such as e-learning or MOOCs benefit from student data that is easily available for analysis and can be used to improve courses [2]. In these settings, student activity takes place in controlled, digitized setups that data can be extracted from by logging interactions [6], [10]. In in-person university courses or other offline educational settings other data collection strategies have to be employed, for example, the results of peer assessments [11] or surveys [1]. In these analog settings, the lack of detailed, high-resolution data on learners can be compensated by collecting additional data sources specific to the individual context, e.g. social network data [12] or even the complexity of teachers' lecture notes [13]. However, due to the limited classroom time available, teachers are often forced to choose between spending time assisting students and spending time assessing students and collecting data [10]. To alleviate this problem, data that is already being created by students during project work,

especially if digital systems are used, can be analyzed in depth. While this type of already existent data has previously been used to assess students [14], [15], [16], it is also valuable to assess the effectiveness of changes in curriculum design and in order to improve classroom courses.

C. Case Study

In this paper, we describe a case study on how software development artifacts, created during students' project work in a software engineering course, can be used to check educators' assumptions on student behavior. The course's setting of collaborative software engineering in a simulated real-world scenario is ideally suited for collecting development data. During the project work, students use common development tools such as version control systems (VCS), issue trackers and Continuous Integration services. The artifacts produced using these systems, i.e. commits in a VCS containing code changes and descriptions, contain a large amount of information on how students work and collaborate in their groups [17], [18].

D. Research Questions

The following research questions (RQ) guide our work:

RQ1 How can surveys be used to gauge students perceptions of changes in course design over time?

RQ2 What data can be collected from software development artifacts created by students during a classroom course?

RQ3 What metrics can be applied to student development data to gauge changes in student behavior during project work?

The rest of the paper is structured as follows: Section II introduces the university course in which the case study was performed and describes the software development process that was followed. Section III presents the surveys that were conducted and discusses the perceptions and attitudes of students. The following Section IV describes the analysis of students' software development artifacts produced in the course installments of the last five years and discusses the results. Section V presents related work in the field of studies in student behaviors and sources of educational data. Section VI concludes and summarizes our findings.

II. Case Study Context

The undergraduate software development course described in this case study has been running in our university for more than 5 years. It is repeatedly run in the winter semester with a length of 15 weeks and was most recently taught in the winter semester of 2017/18.

A. Software Engineering Course

The main goal of the capstone course "Software Engineering II" is teaching iterative, agile development methods and best practices in a hands-on fashion, which has become standard practice in universities [19], [20]. Each week of the course students are expected to work 8 hours on the project including lectures and team meetings. In a simulated real-world scenario, students are encouraged to apply the agile processes, introduced in lectures and exercises, and adapt them to suit their teams. The main learning targets of the course include:

- Gaining experience with the artifacts and meetings of agile methods
- Acquiring knowledge of source code management (SCM) and continuous integration (CI) systems
- Developing critical self-assessment skills regarding students own roles in a software development team

All participants, who form their own development teams, jointly develop a single software system. This means students need to communicate and collaborate within their teams as well as with other student teams. The project is hosted on the public collaboration platform GitHub[1], allowing all stakeholders access to the code and documentation. Junior research assistants, acting as tutors, are present during team meetings and provide advice and assistance. Regular lectures on agile methodologies and their applications as well as more general software development topics, such as Continuous Integration and testing, take place during the course.

B. Course Evolution

Course installments prior to the winter semester of 2014/15 taught exclusively the Scrum methodology [21]. However, Lean development approaches, such as Kanban [22], have gained popularity in industry [23], [24]. Therefore, in an ongoing effort to keep the course as relevant and closely related to real-world scenarios as possible, the practice of Kanban was included. Students employed the Scrum methodology at the beginning of the course, before switching to Kanban. In comparison to Scrum, Kanban is considered less authoritative and prescriptive, having fewer rituals and rules than Scrum [25]. In order to improve learning results, it is therefore advisable to introduce Kanban after students have already gained experience with the more structured Scrum method [26]. Iterations of the course prior to the winter term 2015/16 did not include Kanban and focused solely on the application of Scrum. The inclusion of Kanban in the course is a major change as it impacts how students collaborate, plan their work and organize their team structures and meetings. It needs to be evaluated in order to gauge how effective the introduction of a different software development methodology was, both in terms of student satisfaction as well as how well the method was applied in the project.

C. Switching Development Processes

In the first four development iterations of the course, the majority of the course, a modified version of the Scrum process, adapted for the limited time allotted to students for the course is employed. It is depicted in Figure 1 at the top.

Participants form self-organizing teams [27] of up to 8 members. For every Scrum team the roles of Product Owner (PO) and a Scrum Master (SM), are also performed by student team members, while all other students act as software

[1] https://github.com/

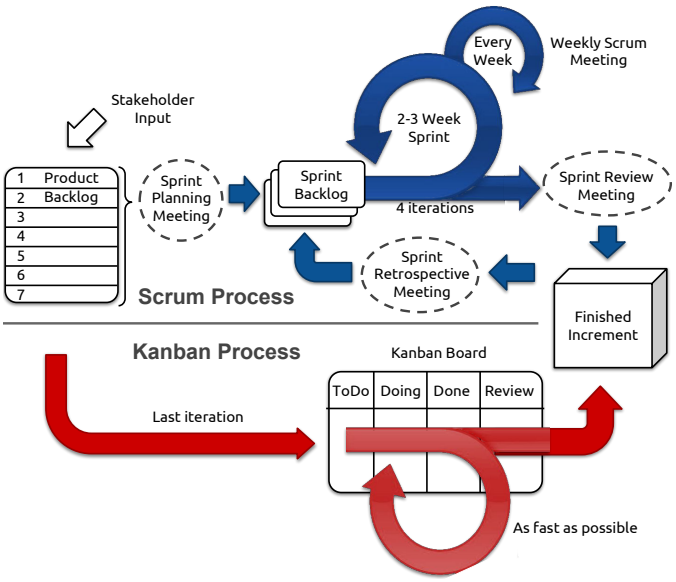

Fig. 1. Overview of the modified Scrum process (top, blue) and the Kanban process (bottom, red) used in our software engineering course.

developers. In every development iteration (i.e. Sprint), a planning, sprint review, retrospective as well as a weekly synchronization meeting, a stand-up meeting, is organized by the teams. After course participants have become familiar with the Scrum process and their teams, i.e. after they have reached the *norming* stage of group development [28], and have developed a cohesive group, Kanban and its practices are introduced in a lecture. The concepts of Kanban such as the Kanban board, the idea of workflow visualization and the guiding principle of limiting work in progress (WIP) [29] are introduced. We encourage students to try out and apply these new ideas in their teams. Participants employ Kanban for the last iteration of the project, instead of a final Scrum sprint, see the bottom of Figure 1.

III. Surveys

When trying to assess the impact of changes in curriculum design and whether the expected changes to student behavior took place, students' perceptions can be collected through the use of surveys.

A. End-of-term Survey

As part of an ongoing effort to collect feedback from students to improve teaching and university courses, standardized end-of-term surveys were conducted in all iterations of our software engineering courses in the years 2013 up to 2018. This has become standard practice for educational institutions to evaluate teaching quality [1]. The survey is administered online before students receive their final course grades to prevent interference. The survey collects perceptions of students on a range of topics, including satisfaction with the course in general, perceived importance of course contents and satisfaction with mentoring. An extract of the questions relevant to student satisfaction with the project work and the course over the iterations of the course is shown in Table I.

#	Survey item
1	The course was fun
2	The course motivated me to delve deeper into the discussed topics
3	I learned a lot in the course
4	The course is important to my course of studies
5	The course was well structured
6	The topics of the course were well chosen
7	How would you rate the course overall?

TABLE I
Questions and statements of the end-of-term survey.

Questions and statements could be rated on a scale of "fully agree"/"great" to "totally disagree"/"bad". Results of the anonymous survey are presented to course instructors in aggregate form, with ratings mapped to German school grades. The grade 1 ("very good") is the best, with the grade 5 ("inadequate") signifying a fail.

Results Average ratings for all installments of the course showed overwhelmingly positive perceptions of the course and its content, see Figure 2. Very few questions were answered

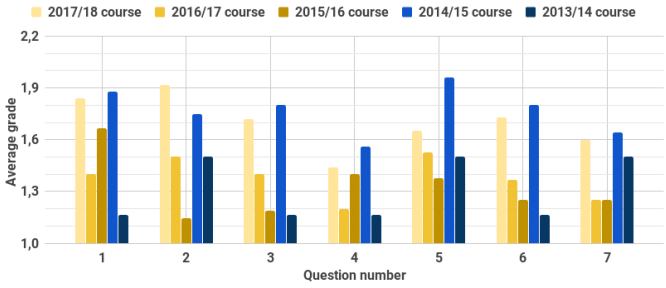

Fig. 2. Mean grades given to course installments by students after the course's end. German school grades: 1 is the best grade, i.e. the lower the better. Courses in 2013/14 and 2014/15 (blue) employed solely Scrum, the others (yellow) employed both Scrum and Kanban.

with mean scores larger than 2 ("good"). While this is satisfying to see as far as student satisfaction goes it also means no significant change can be detected in student satisfaction between the courses employing Kanban and those that did not. The variance in answers is very low, as students rated aspects of the course overwhelmingly as very good (1) to good (2). Therefore, we devised a more specific survey.

B. Kanban Survey

In the second installment of the software engineering course that used Kanban (in 2016/17), we conducted a voluntary, anonymous online survey among all students after course completion, in addition to the regular end-of-term surveys. The first course that introduced Kanban (in 2015/16) using newly created teaching materials had received critical comments from students in oral feedback sessions, which we addressed in the following course. The Kanban survey focused on students' perceptions of Kanban as well as the advantages and drawbacks

of the introduced practices and methods. While the survey was designed to elicit responses to the details of how Kanban was introduced in the course, it also explicitly included questions on whether and how students' workflows were adapted when changing from Scrum to Kanban processes. These questions were designed to better understand whether the expected changes in process had actually taken place. The survey questions related to process change are listed in Table II.

#	Type	Question
1	5-point scale	Was the Kanban week at project end more useful and productive then a last week of Scrum?
2	5-point scale	Did you have to adapt your workflow for the Kanban week?
3	free text	What were the biggest advantages and disadvantages of using Kanban in your team?
4	multiple choice	How did user stories change from using Scrum to Kanban?
5	5-point scale	Would you recommend using Kanban to the participants of next year's course?

TABLE II
QUESTIONS RELATED TO KANBAN ADOPTION OF THE ANONYMOUS ONLINE STUDENT SURVEY PERFORMED AT THE END OF THE 2016/17 SOFTWARE ENGINEERING COURSE.

The survey consisted mainly of questions that could be answered using a 5-point Likert scale, ranging from 1 (strong no) to 5 (strong yes), with 3 being neutral. Additionally, the survey included free text questions as well as a multiple choice question t gather more detailed insights. It was possible to submit the survey with missing answers.

Results Overall, 18 students, 17 men and 1 woman, answered the questionnaire. All questions featuring the Likert scale were answered by all participants. Table III contains a summary of the collected answers. Concerning the change of Scrum to Kanban methods (question 2), students on average stated that they had adapted their workflows, see Figure 3. The high mean value (4.08), as well as the median of 5 (highest agreement), point to students having adapted their workflow in a reflected manner. The overall positive student attitude towards group software development methodologies was also

#	Question Topic	Mean	Std. Dev.	10% Trim. Mean	Median	Range
1	Kanban week preferred over another Scrum week?	4.08	1.38	4.30	5.00	4.00
2	Was the workflow adapted?	3.83	1.11	4.00	4.00	4.00
5	Recommended for next year?	4.33	0.98	4.50	5.00	3.00

TABLE III
SUMMARIZED ANSWERS OF PARTICIPANTS TO THE 5-POINT LIKERT SCALE QUESTIONS OF THE SURVEY. ANSWER POSSIBILITIES: 1 (STRONG NO), 2 (NO), 3 (NEUTRAL), 4 (YES) 5 (STRONG YES).

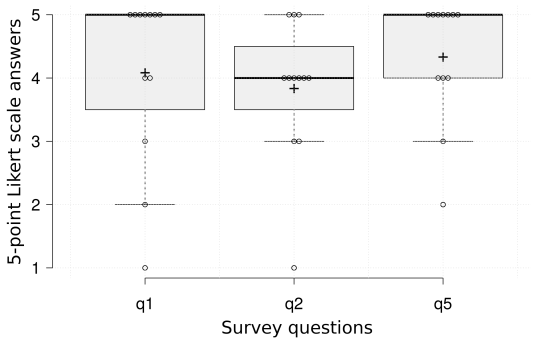

Fig. 3. Summary of answers to Likert-scale questions 1, 2 and 5 as a box plot. Center lines show the medians, box limits indicate the 25th and 75th percentiles. Whiskers extend 1.5 times the interquartile range from the 25th and 75th percentiles, outliers are represented by dots. Crosses represent sample means, data points are plotted as open circles. N = 12.

reflected in the answers to questions 1 and 6, regarding the preference of Kanban over Scrum for the last iteration as well as recommending the course for next year's students. Surveys participants indicated that they would strongly recommend the usage of Kanban to the next cohort of students of the software engineering course (question 6), indicating that, even though this question is not a measure of learning success, applying Kanban was most likely at least fun.

The free text answers to question 3, see Figure II, regarding the (dis)advantages of Kanban, were manually labeled with the mentioned topics. The list of topics was refined repeatedly after evaluating every question. Survey participants identified the following topics as advantages of Kanban (N=11):

- Efficiency (7 mentions)
- Autonomy (4 mentions)

Three other other topics were mentioned only twice or fewer times. As concepts, efficiency and autonomy are closely related to Lean Software's guiding principles of *Eliminate Waste* and *Empowering the Team*, respectively [30]. As Kanban is heavily inspired by these ideas, it is reassuring to see that these ideas transferred.

Regarding the disadvantages of Kanban usage, students mentioned the following topics (N=9):

- Only worked on small user stories (3)
- Uneven task distribution (2)

Another six disadvantages only received single mentions.

Solely working on small user stories, i.e. work items in an agile process, may be a consequence of team member autonomy. Developers may choose to work on small items that can be moved through the columns of the Kanban board quickly, instead of picking larger, more time-consuming tasks to work on. Tackling uneven task distribution between team members is an ongoing challenge in educational settings and especially in self-organizing teams of students. It can be seen as a negative consequence of the autonomy identified by survey participants. Developers are free to handle their workload, with some developers choosing to do more and others choosing to work on fewer items.

Topic	Answer choice and count			
User story focus	bug-oriented	11	feature-oriented	0
User story length	Shorter	11	Longer	0
Requirements	More detailed	8	More general	0
Interaction with PO	More	3	Less	0
Prioritization of stories	Better	3	Worse	2

TABLE IV
ANSWERS OF SURVEY PARTICIPANTS TO QUESTION #4, REGARDING ATTRIBUTES OF USER STORIES WHEN CHANGING FROM SCRUM TO KANBAN PROCESSES. N=12.

User stories are one of the core means of communication, both in Kanban and Scrum, between the Product Owner, who receives input from stakeholders, and developers [31]. As such, we included a question on the perceived change of user stories when switching from Scrum to Kanban (question 4). In order to make answering easier, this question was a multiple choice question that provided a range of answer possibilities of which any number could be chosen. The choices, as well as the summarized answers of survey participants, are shown in Table IV.

Students classified the user stories that were written by Product Owners and developers during the Kanban iteration as shorter and more bug-oriented than in the previous Scrum iterations. While an influx of small fixes to a software product is expected shortly before the final deadline, a time in which Kanban was used, smaller user stories can also help move tickets through the Kanban board more quickly. This reduced the *cycle time*, the time from when work begins on an item until it is ready for delivery [32].

However, students also answered that the requirements, i.e. part of the *acceptance criteria* [33] within user stories, had gotten more detailed during Kanban usage. This allows work on a user story to be started without having to clarify open questions beforehand and can help efficiency by decreasing the cycle time. Small user stories that contain enough detail to be immediately implementable are ideal for usage in the Kanban process [34].

C. Discussion

Both surveys, the more general end-of-term survey as well as the more specialized survey on Kanban, revealed positive attitudes towards our approach of teaching agile processes in a hands-on fashion as well as the shift from Scrum to Kanban at the end of the project (*RQ2*). Students indicated that they would recommend using Kanban to participants of the following year's course. These results are in line with related, similar studies [35]. In particular, Melnik et al. state that students, in general, were "very enthusiastic about core agile practices" and accepted and liked them [36]. The authors also point out that this observation held for a broad range of students, regardless of educational program, age or industry experience. Furthermore, whether the educational setting that student teams worked in during their project work was more or less controlled, did not have a significant influence on student satisfaction [37]. While the findings of this and similar studies are encouraging for educators teaching software project courses, they also pose challenges. If students are generally content in agile project courses, simply due to the course setting and the fact that teamwork and building software together is fulfilling, how can improvements to curriculum design and changes in student behavior be evaluated?

Student opinions and attitudes towards course contents are an important part of any assessment plan. However, evaluations of teaching methods should primarily rely on the assessment of learning outcomes [1], which surveys only partly capture as they are geared towards collecting perceptions and attitudes. Furthermore, if not every survey participant answered the more reflective, time-consuming free text answers, data on the learning outcomes of these participants is missing completely. To help tackle these challenges, the outcomes and artifacts produced during the process transition from Scrum to Kanban can be analyzed to gain additional insights into development teams.

IV. DEVELOPMENT ARTIFACT ANALYSIS

While surveys are effective tools for capturing the attitudes of participants, they do not allow insights into whether the perceived change in workflow or in user story quality actually took place during the project or how severe the change was. In order to provide another dimension of analysis based on real project data, we evaluated the software development artifacts produced by course participants. In particular, we compared students' development artifacts of the last five installments of our software engineering course, the earliest two of which did not include Kanban and the three most recent ones that did.

A. Data Collection

The development artifacts we collected from course repositories included commits into the version control system *git*[2], containing code changes, timestamps and commit messages describing the change, as well as tickets, acting as user stories, in an issue tracker. Both of these data sources were collected from the collaboration and hosting service GitHub, where all projects were hosted. GitHub features extensive application programming interfaces (APIs) [3] that allow programmatically extracting the data stored by the service.

For every repository of the last five course iterations, user stories/issues and commits from the last seven days of project work were collected. This is the time frame that Kanban was employed in the more recent course iterations.

1) Contributors: First, all unique contributors of a project in the given time frame were identified. This allows information extracted from different courses to be normalized by contributor count, as different course installments featured differing participant amounts. Manual deduplication of users followed this step, to merge accounts where students had used multiple accounts or email addresses, e.g. university or private accounts, to work on projects.

[2]https://git-scm.com/
[3]https://developer.github.com/v3/

2) Issues / User Stories: Issues were only included in the analysis if they were closed in the study time frame and issues that represented GitHub pull requests[4] and not user stories were excluded. For every issue, the number of comments, as well as events[5], interactions except commenting, such as assigning developers or labels, were collected. We furthermore annotated each issue with whether the user who had opened the issue was the same as the one who closed it.

3) Commits: In order to allow more rapid analysis, the git repositories of all projects under study where copied, i.e. *cloned*, to a local copy. Using the git command line, specifically the *git log* sub command[6], statistics on the commits where collected. Attention was paid not to include merge commits, i.e. commits that introduce no new functionality and to take advantage of the deduplicated list of contributors. All statistics were collected as means per contributor. For every list of commits of a project repository, the mean commits, touched files, last-minute commits, mean line changes and the number of unique issues referenced, were saved. Last-minute commits refer to those commits made within a day of project end. Furthermore, we parsed the commit messages of commits and identified whether they referenced an issue in the issue tracker by number in the form "fixed issue #123".

Using this data, both the assumptions on student behavior, i.e. educators hypothesis of how artifacts would change from using Scrum to Kanban, as well as the accurateness of student perceptions could be tested.

B. Discussion

The collected data shows that the length of user stories did not significantly differ from when Kanban or Scrum was used in the last iteration of the course, see Table V.

	Issue body length			Issue title length		
Course year	**Mean**	**Stdev**	**Median**	**Mean**	**Stdev**	**Median**
2013/14	274.8	295.2	169.0	35.3	15.3	32.0
2014/15	420.8	327.3	361.0	50.7	15.5	50.0
2015/16*	360.9	339.4	253.0	36.9	16.9	32.5
2016/17*	505.5	556.9	378.0	36.9	16.7	35.0
2017/18*	579.8	393.3	526.0	35.9	14.3	37.5

TABLE V
ISSUE BODY AND TITLE LENGTH OF ISSUES FOR THE LAST WEEK OF PROJECTS. COURSES MARKED WITH * EMPLOYED KANBAN.

This differs from the reported perceptions of students in the survey performed in the 2016/17 course installment. There, students reported that user stories were perceived to be shorter when using Kanban when compared to Scrum. While these two measures are not necessarily directly comparable, further study into the content differences between user stories in Kanban and Scrum is required. However, the analysis was able to uncover this discrepancy and provides a starting point for further investigation.

[4]https://help.github.com/articles/about-pull-requests/
[5]https://developer.github.com/v3/issues/events/
[6]https://git-scm.com/docs/git-log

Most other measures calculated from commits, such as the mean amount of touched files, did not differ significantly between the two processes in different course years, see Table VI.

Course year	**Commit amount**	**Touched files**	**Last-minute commits**	**Line changes per commit**	**Unique issues referenced**
2013/14	12.1	13.3	1.4	590.5	2.8
2014/15	7.2	6.9	1.2	408.0	1.0
2015/16*	6.1	8.0	8.1	466.0	0.1
2016/17*	3.4	5.4	2.2	163.5	2.2
2017/18*	8.6	5.3	1.7	195.0	1.5

TABLE VI
COMPARISON OF COMMIT ATTRIBUTES FOR THE LAST WEEK OF PROJECTS. COURSES MARKED WITH * EMPLOYED KANBAN. ALL VALUES STATED NORMALIZED BY COURSE PARTICIPANT COUNT.

However, it is encouraging to see that the amount of *last-minute commits*, i.e. commits made close to the end of the iteration [38] tended to be higher in the course installments in which Kanban was employed. Scrum's iteration plan, the Sprint Backlog, contains all the work items a team intends to address in a sprint, i.e. the amount of work that can be performed by a team in an iteration. Ideally, these items are worked on in a continuous manner, so that towards the end of the sprint the last user story is finished [21]. In this manner, work intensity and commit frequency should be uniformly distributed during an iteration. In contrast, Kanban does not explicitly call for iteration planning and so work is more likely to be assigned more dynamically: new work items can easily be added to the work queue, especially towards the end of the project, when the deadline approaches.

The more dynamic nature of Kanban is also reflected in the fact that the mean line change per commit, as well as the mean number of touched files, were smaller in the last two years of the course, see Table VI, when educators had already gathered some experience teaching the new methodology. This is in line with the Kanban survey results, where students stated that their user stories when employing Kanban were more bug-oriented than feature-oriented. Fixing a bug usually requires changing fewer lines touching fewer files than implementing an entirely new feature. Furthermore, bugs are usually noticed during regular development activities or during testing and can easily be added to a Kanban board [39], whereas they might only end up in the next Scrum sprint [40].

Interactions with user stories, i.e. issues on GitHub, did not differ significantly between those courses that employed kanban and those that used Scrum, see Table VII.

Normalized by participant count, similar mean numbers of issues were closed in the studied time frame and a similar amount of comments were attached to issues. However, the two most recent courses using Kanban showed higher mean amounts of non-comment events, such as labeling or assignments, a sign that the issue tracker was used more heavily. While Scrum explicitly calls for a role that mainly writes and

Course year	Mean per contributor			% issues opened & closed by same person
	Issue amount	Issue events	Issue comments	
2013/14	4.9	27.5	17.1	43
2014/15	2.6	47.8	4.1	58
2015/16*	3.9	35.5	4.1	20
2016/17*	2.8	64.4	6.4	46
2017/18*	2.1	68.9	4.8	65

TABLE VII
COMPARISON OF ISSUES AND THEIR ATTRIBUTES FOR THE LAST WEEK OF PROJECTS. COURSES MARKED WITH * EMPLOYED KANBAN.

prioritizes user stories, the Product Owner [21], Kanban does not. We had thus hypothesized that introducing Kanban would result in higher engagement by the entire team with the list of outstanding work items, instead of teams mostly relying on the PO to maintain it. However, the percentage of issues opened and closed by the same person, see the last column of Table VII, was surprisingly low even in the Scrum courses, with only 43% and 58% in course installments 2013/14 and 2014/15. These numbers did not differ significantly for the Kanban courses. While these results do not support our original hypothesis on team engagement with user stories, they represent opportunities for future work on how agile student teams interact with user stories and work item backlogs in collaboration with a Product Owner role.

By analyzing software development data created by students, which is already produced during regular development activities, we were able to uncover areas where some of our assumptions on student behavior regarding the adoption of different agile software development methodologies were confirmed and some were refuted. These areas will serve as a basis for future improvements to the course.

V. RELATED WORK

A variety of specialized data sources have been analyzed in previous work in order to gain insights into student behaviors in computer science courses.

Wilson and Shrock [41] employed a survey to determine factors that promote success in an introductory computer science course. They examined twelve factors of which comfort level, math, and attribution to luck for success/failure were the most important for predicting student scores. Similarly, Bennedsen and Caspersen [42] attempted to help improve students' learning premises by collecting data on the emotional and social factors of students in a survey. They collected data on the factors of perfectionism, self-esteem, coping tactics, affective states, and optimism.

While surveys can be used to collect data on arbitrary factors, depending on which questions are used, entirely different ways of collecting student data have also been explored. Keen and Etzkorn [13] analyzed the complexity of teachers' lecture notes, the *buzzword density*, i.e. the amount of Computer Science domain-specific words divided by the total number of words in the lecture, to predict grades. Fire et al. [12] built a graph of social interaction between students using homework assignment data and course website logs. The authors then studied this data structure to reveal the impact of cooperation among students on their success. The model showed a high correlation between a student's grade and that of their closest friend. Ashenafi et al. used data created by students during course activities, namely the results of several semi-automated peer-assessment tasks throughout the semester, to build a linear regression model for predicting final grades [6].

Performance prediction has more recently also been applied in MOOCs, which feature an abundance of digital data on student behavior. Jian et al. [43] used students' activity in discussion forums together with their performance on peer-assessment tasks and assignments in the first week of the course to predict whether students will complete the course. Brinton et al. [44] collected behavior data of students watching the educational videos of a MOOC. They recorded interactions with the video player, e.g. pause, rate change or seek in order to predict whether participants will correctly answer questions on a video's content in the first attempt. Teusner et al. [8] compiled events of users interacting with a MOOC platform in the form of standardized events [45]. From these event streams 17 combined metrics, such as session duration, forum, video player and download activity, as well item discovery are computed. The authors state that using the collected data *informed actions* can be taken to improve learning outcomes.

VI. CONCLUSION

University courses need to be continuously adapted to changes in requirements, such as industry shifts, technology advancements or altered student expectations. Feedback to educators on curriculum design usually takes the form of end-of-term surveys or questionnaires on specific course aspects. However, techniques from the field of Educational Data Mining can be employed to gain insights into student teams and their reactions to changes in curricula that go beyond those possible with surveys. This paper presents our approach to analyze development artifacts to achieve this goal. We analyzed the software development artifacts of student teams from five university undergraduate software development courses, which teach different agile development methodologies. Surveys with participants revealed positive attitudes towards the course and changing the employed development methodology during the course from Scrum to Kanban. However, surveys were not able to ascertain the degree to which students had adapted their workflows accurately. Therefore, we propose an approach of analyzing the software development data created by students during regular project work activities, specifically user stories and commits to a version control system, as an additional dimension of analysis. While this data serves a primary purpose in communication between students it also represents information that can be collected and analyzed by educators to generate insights into student behavior during project work.

References

[1] R. M. Felder and R. Brent, "How To Improve Teaching Quality," *Quality Management Journal*, vol. 6, no. 2, pp. 9–21, 2009.

[2] A. Dutt, M. A. Ismail, and T. Herawan, "A Systematic Review on Educational Data Mining," *IEEE Access*, vol. 5, no. c, pp. 15 991–16 005, 2017. [Online]. Available: https://doi.org/10.1109/ACCESS.2017.2654247

[3] K. R. Koedinger, K. Cunningham, A. Skogsholm, and B. Leber, "An open repository and analysis tools for fine-grained, longitudinal learner data," in *Educational Data Mining 2008: 1st International Conference on Educational Data Mining, Proceedings*, 2008, pp. 157–166.

[4] C. Romero and S. Ventura, "Educational Data Mining: A Review of the State of the Art," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 6, pp. 601–618, nov 2010. [Online]. Available: https://doi.org/10.1109/TSMCC.2010.2053532

[5] International Educational Data Mining Society, "educationaldatamining.org," 2018. [Online]. Available: http://educationaldatamining.org/

[6] M. M. Ashenafi, G. Riccardi, and M. Ronchetti, "Predicting students' final exam scores from their course activities," in *2015 IEEE Frontiers in Education Conference (FIE)*. IEEE, oct 2015, pp. 1–9. [Online]. Available: https://doi.org/10.1109/FIE.2015.7344081

[7] J. Gamulin, O. Gamulin, and D. Kermek, "Data mining in hybrid learning: Possibility to predict the final exam result," in *2013 36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2013, pp. 591–596.

[8] R. Teusner, K.-A. Rollmann, and J. Renz, "Taking Informed Action on Student Activity in MOOCs," in *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale - L@S '17*. New York, New York, USA: ACM Press, 2017, pp. 149–152. [Online]. Available: https://doi.org/10.1145/3051457.3053971

[9] C. Romero, S. Ventura, and P. D. Bra, "Knowledge Discovery with Genetic Programming for Providing Feedback to Courseware Authors," *User Modeling and User-Adapted Interaction*, vol. 14, no. 5, pp. 425–464, jan 2004. [Online]. Available: https://doi.org/10.1007/s11257-004-7961-2

[10] M. Feng and N. T. Heffernan, "Informing Teachers Live about Student Learning: Reporting in the Assistment System," in *The 12th Annual Conference on Artificial Intelligence in Education Workshop on Usage Analysis in Learning Systems*, 2005.

[11] K. J. Topping, "Methodological quandaries in studying process and outcomes in peer assessment," *Learning and Instruction*, vol. 20, no. 4, pp. 339–343, aug 2010. [Online]. Available: https://doi.org/10.1016/j.learninstruc.2009.08.003

[12] M. Fire, G. Katz, Y. Elovici, B. Shapira, and L. Rokach, "Predicting Student Exam's Scores by Analyzing Social Network Data," in *Active Media Technology*, R. Huang, A. A. Ghorbani, G. Pasi, T. Yamaguchi, N. Y. Yen, and B. Jin, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 584–595.

[13] K. J. Keen and L. Etzkorn, "Predicting Students' Grades in Computer Science Courses Based on Complexity Measures of Teacher's Lecture Notes," *Journal of Computing Sciences in Colleges*, vol. 24, pp. 44–48, 2009.

[14] P. Johnson, Hongbing Kou, J. Agustin, Qin Zhang, A. Kagawa, and T. Yamashita, "Practical automated process and product metric collection and analysis in a classroom setting: lessons learned from Hackystat-UH," in *Proceedings. 2004 International Symposium on Empirical Software Engineering, 2004. ISESE '04.* IEEE, aug 2004, pp. 136–144. [Online]. Available: https://doi.org/10.1109/ISESE.2004.1334901

[15] C. Matthies, T. Kowark, K. Richly, M. Uflacker, and H. Plattner, "How surveys, tutors, and software help to assess Scrum adoption in a classroom software engineering project," in *Proceedings of the 38th International Conference on Software Engineering Companion (ICSE)*. New York, New York, USA: ACM Press, 2016, pp. 313–322. [Online]. Available: https://doi.org/10.1145/2889160.2889182

[16] N. Zazworka, K. Stapel, E. Knauss, F. Shull, V. R. Basili, and K. Schneider, "Are developers complying with the process," in *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM '10*, ACM. New York, New York, USA: ACM Press, 2010, p. 1. [Online]. Available: https://doi.org/10.1145/1852786.1852805

[17] C. Rosen, B. Grawi, and E. Shihab, "Commit guru: analytics and risk prediction of software commits," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2015*, ser. ESEC/FSE 2015. New York, New York, USA: ACM Press, 2015, pp. 966–969. [Online]. Available: https://doi.org/10.1145/2786805.2803183

[18] E. A. Santos and A. Hindle, "Judging a commit by its cover," in *Proceedings of the 13th International Workshop on Mining Software Repositories - MSR '16*, ser. MSR '16. New York, New York, USA: ACM Press, 2016, pp. 504–507. [Online]. Available: https://doi.org/10.1145/2901739.2903493

[19] M. Paasivaara, J. Vanhanen, V. T. Heikkilä, C. Lassenius, J. Itkonen, and E. Laukkanen, "Do High and Low Performing Student Teams Use Scrum Differently in Capstone Projects?" in *Proceedings of the 39th International Conference on Software Engineering: Software Engineering and Education Track*, ser. ICSE-SEET '17. Piscataway, NJ, USA: IEEE Press, 2017, pp. 146–149. [Online]. Available: https://doi.org/10.1109/ICSE-SEET.2017.22

[20] D. Dzvonyar, L. Alperowitz, D. Henze, and B. Bruegge, "Team Composition in Software Engineering Project Courses," *SEEM'18: IEEE/ACM International Workshop on Software Engineering Education for Millennials*, 2018.

[21] K. Schwaber and J. Sutherland, "The Scrum Guide - The Definitive Guide to Scrum: The Rules of the Game," Tech. Rep., 2017. [Online]. Available: http://scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf

[22] J. P. Womack, D. T. Jones, and D. Roos, *The machine that changed the world: the story of lean production*. Harper Collins, 1991.

[23] VersionOne Inc., "The 11th Annual State of Agile Report," VersionOne Inc., Tech. Rep., 2017. [Online]. Available: https://explore.versionone.com/state-of-agile/versionone-11th-annual-state-of-agile-report-2

[24] A. Komus and M. Kuberg, "Abschlussbericht : Status Quo Agile 2016/2017," Hochschule Koblenz, University of Applied Sciences, Tech. Rep., 2017.

[25] H. Kniberg, *Kanban and Scrum - Making the most of both*. Lulu.com, 2009.

[26] V. Mahnic, "From Scrum to Kanban: Introducing Lean Principles to a Software Engineering Capstone Course," *International Journal of Engineering Education*, vol. 31, no. 4, pp. 1106–1116, 2015.

[27] R. Hoda, J. Noble, and S. Marshall, "Organizing self-organizing teams," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - ICSE '10*, vol. 1. New York, New York, USA: ACM Press, 2010, p. 285. [Online]. Available: https://doi.org/10.1145/1806799.1806843

[28] D. A. Bonebright, "40 years of storming: a historical review of Tuckman's model of small group development," *Human Resource Development International*, vol. 13, no. 1, pp. 111–120, feb 2010. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/13678861003589099

[29] M. O. Ahmad, J. Markkula, and M. Oivo, "Kanban in software development: A systematic literature review," in *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*. IEEE, sep 2013, pp. 9–16. [Online]. Available: https://doi.org/10.1109/SEAA.2013.28

[30] M. Poppendieck and T. Poppendieck, *Lean software development: an agile toolkit*. Addison-Wesley, 2003.

[31] M. Rees, "A feasible user story tool for agile software development?" in *Ninth Asia-Pacific Software Engineering Conference, 2002.*, vol. 2002-Janua. IEEE Comput. Soc, 2002, pp. 22–30. [Online]. Available: https://doi.org/10.1109/APSEC.2002.1182972

[32] R. Polk, "Agile and Kanban in Coordination," in *2011 AGILE Conference*. IEEE, aug 2011, pp. 263–268. [Online]. Available: https://doi.org/10.1109/AGILE.2011.10

[33] A. Silva, T. Araújo, J. Nunes, M. Perkusich, E. Dilorenzo, H. Almeida, and A. Perkusich, "A systematic review on the use of Definition of Done on agile software development projects," in *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering - EASE'17*. New York, New York, USA: ACM Press, 2017, pp. 364–373. [Online]. Available: https://doi.org/10.1145/3084226.3084262

[34] N. Nikitina and M. Kajko-Mattsson, "Developer-driven big-bang process transition from Scrum to Kanban," in *Proceeding of the 2nd workshop on Software engineering for sensor network applications - SESENA '11*. New York, New York, USA: ACM Press, 2011, p. 159. [Online]. Available: https://doi.org/10.1145/1987875.1987901

[35] V. Mahnič, "Scrum in software engineering courses: An outline of the literature," *Global Journal of Engineering Education*, vol. 17, no. 2, pp. 77–83, 2015.

[36] G. Melnik and F. Maurer, "A cross-program investigation of students' perceptions of agile methods," in *Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005.* IEEE, may 2005, pp. 481–488. [Online]. Available: https://doi.org/10.1109/ICSE.2005.1553593

[37] C. Matthies, T. Kowark, and M. Uflacker, "Teaching Agile the Agile Way Employing Self-Organizing Teams in a University Software Engineering Course," in *American Society for Engineering Education (ASEE) International Forum*. New Orleans, Louisiana: ASEE, 2016. [Online]. Available: https://peer.asee.org/27259

[38] C. Matthies, T. Kowark, M. Uflacker, and H. Plattner, "Agile metrics for a university software engineering course," in *IEEE Frontiers in Education Conference (FIE)*. Erie, PA: IEEE, oct 2016, pp. 1–5. [Online]. Available: https://doi.org/10.1109/FIE.2016.7757684

[39] M. Ikonen, E. Pirinen, F. Fagerholm, P. Kettunen, and P. Abrahamsson, "On the Impact of Kanban on Software Project Work: An Empirical Case Study Investigation," in *2011 16th IEEE International Conference on Engineering of Complex Computer Systems*. IEEE, apr 2011, pp. 305–314. [Online]. Available: https://doi.org/10.1109/ICECCS.2011.37

[40] H. Kniberg, *Scrum and XP from the Trenches*. C4Media, 2007.

[41] B. C. Wilson and S. Shrock, "Contributing to success in an introductory computer science course," *ACM SIGCSE Bulletin*, vol. 33, no. 1, pp. 184–188, mar 2001. [Online]. Available: https://doi.org/10.1145/366413.364581

[42] J. Bennedsen and M. E. Caspersen, "Optimists have more fun, but do they learn better? On the influence of emotional and social factors on learning introductory computer science," *Computer Science Education*, vol. 18, no. 1, pp. 1–16, 2008.

[43] S. Jiang, A. E. Williams, K. Schenke, M. Warschauer, and D. O. Dowd, "Predicting MOOC Performance with Week 1 Behavior," *Proceedings of the 7th International Conference on Educational Data Mining (EDM)*, pp. 273–275, 2014.

[44] C. G. Brinton and M. Chiang, "MOOC performance prediction via clickstream data and social learning networks," in *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, apr 2015, pp. 2299–2307. [Online]. Available: https://doi.org/10.1109/INFOCOM.2015.7218617

[45] A. del Blanco, A. Serrano, M. Freire, I. Martinez-Ortiz, and B. Fernandez-Manjon, "E-Learning Standards and Learning Analytics," *2013 Ieee Global Engineering Education Conference*, pp. 1255–1261, 2013. [Online]. Available: https://doi.org/10.1109/EduCon.2013.6530268